Improving Adaptation Knowledge Discovery by Exploiting Negative Cases: First Experiment in a Boolean Setting

Tristan Gillard, Jean Lieber, Emmanuel Nauer

Improving adaptation knowlege discovery by exploiting negative cases

Tristan Gillard, Jean Lieber, and Emmanuel Nauer

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract. Case-based reasoning usually exploits *positive* source cases consisting in a source problem and its solution that is known to be a correct for the problem. The work presented in this paper addresses in addition of positive case exploitation, the exploitation of *negative* cases, i.e. problem-solution pairs where the solution is an incorrect answer to the problem, which can be acquired when the case-based reasoning (CBR) process fails. An originality of this work is that positive and negative cases are used both for adaptation knowledge (AK) discovery using closed itemsets built on variations between cases. Experiments show that exploiting negative cases in addition to positive ones improves the quality of the AK being extracted and, so, improves the results of the CBR system.

Keywords: adaptation knowledge discovery, closed itemset extraction, negative cases, case-based reasoning

1 Introduction

Case-based reasoning (CBR) [14] aims at solving a new problem—the *target problem*—thanks to a set of cases (the *case base*), where a case is a pair consisting of a problem and a solution to this problem. A *source case* is a case from the case base, consisting of a *source problem* and one of its solutions. The classical approach to CBR consists of selecting source cases *similar* to the target problem and adapting them to solve it. The adaptation step may use different approaches, one of them is the use of adaptation knowledge (AK). Acquiring AK is, in this case, a crucial issue.

Most of the times, AK discovery for CBR focuses on the exploitation of *positive* source cases [5, 3, 2, 7]. A *positive* source case is a source case such that the solution is a correct solution to the problem. However, a case base may also contains *negative* source cases. A *negative* source case is a source case such that the solution is an incorrect solution to the problem. Such negative cases can for example be acquired at the retain step of the classical 4R (retrieve, reuse, revise, retain) CBR process [1], when the CBR process fails and returns an incorrect solution.

This paper presents an approach exploiting at the best all the existing cases of the case base, the positive ones but also the negative ones, in order to improve the AK discovery, under the hypothesis that better the AK quality is, better the results of the CBR system that will use it will be. This work is based on the approach proposed in [2] for extracting AK. The approach is based on closed itemset (CI) extraction on variations between cases. The originality of our work lies in the use of CI extraction to take into account negative cases.

The paper is organized as follows. Section 2 introduces the motivations and the preliminaries for this work, introducing CI extraction and CBR with related work. Section 3 describes our approach for exploiting positive and negatives cases in an AK discovery process. Section 4 presents the evaluation of our approach through experiments and discusses the results. Section 5 points out lines for future research.

2 Motivation and preliminaries

Cooking CBR systems, as the ones which have participated to the *Computer Cooking Contest* (e.g. TAAABLE [4]) are typical of systems that are concerned by the objective of this work. Indeed, it has been showed that adaptating cooking recipes takes benefit from the use of AK [7]. Moreover, feedback may be collected about the system results. For example, the TAAABLE system provides a result interface allowing the users to evalauate whether a recipe adaptation is correct or not [15]. This approach to manage correct and uncorrect adaptations is a way to collect positive and negative cases, which can both be stored in the case base (with an appropriate label).

2.1 Itemset extraction

Itemset extraction is a collection of data-mining methods for extracting regularities into data, by aggregating object items appearing together. Like FCA [10], itemset extraction algorithms start from a *formal context* K, defined by $K = (G, M, I)$, where G is a set of objects, M is a set of items, and I is the relation on $G \times M$ stating that an object is described by an item [10]. Table 1 shows an example of context, in which 4 recipes are described by the ingredients they require: G is a set of 4 objects (recipes r^1, r^2, r^3, and r^4), M is a set of 5 items (ingredients `Apple`, `PieCrust`, `PuffPastry`, `Sugar`, and `Cream`).

An *itemset* I is a set of items, and the *support* of I, $\texttt{supp}(I)$, is the number of objects of the formal context having every item of I. I is frequent, with respect to a threshold σ, whenever $\texttt{supp}(I) \geq \sigma$. I is closed if it has no proper superset J ($I \subsetneq J$) with the same support. For example, $\{\texttt{Apple}, \texttt{PieCrust}\}$ is an itemset and $\texttt{supp}(\{\texttt{Apple}, \texttt{PieCrust}\}) = 2$ because exactly 2 recipes require both `Apple` and `PieCrust`. However, $\{\texttt{Apple}, \texttt{PieCrust}\}$ is not CI, because $\{\texttt{Apple},$

	Apple	PieCrust	PuffPastry	Sugar	Cream
r^1	×	×		×	
r^2	×	×		×	×
r^3			×	×	×
r^4	×		×		

Table 1: An example of formal context representing ingredients used in recipes.

	Apple$^-$	Apple$^=$	Apple$^+$	PieCrust$^-$	PieCrust$^=$	PieCrust$^+$	PuffPastry$^-$	PuffPastry$^=$	PuffPastry$^+$	Sugar$^-$	Sugar$^=$	Sugar$^+$	Cream$^-$	Cream$^=$	Cream$^+$
V^{R^1,R^2}		×			×						×				×
V^{R^1,R^3}	×			×				×			×			×	
V^{R^1,R^4}		×		×				×	×				×		

Table 2: Formal context for ingredient variations in pairs of recipes (r^1, r^2), (r^1, r^3) and (r^1, r^4).

PieCrust, Sugar$\}$ has the same support. For $\sigma = 2$, the frequent CIs (FCIs) of this context are $\{$Apple, PieCrust, Sugar$\}$ and $\{$Sugar, Cream$\}$.

For our experiments, we use CORON [16] a software platform which implements efficient algorithms for symbolic data mining and especially FCI computation.

2.2 Exploiting case variations for AK discovery

Exploiting case variations is not a new idea. [11] introduces this approach of AK learning based on pairwise comparisons of cases. This approach has been applied in various domains such as chemistry [3], medicine [5] or cooking [2, 7].

For an ordered pair of cases $(\mathtt{c}^1, \mathtt{c}^2)$, the approach consists in representing what features have to be removed $(-)$, kept $(=)$ and added $(+)$ to transform $\mathtt{c}^1$ into $\mathtt{c}^2$. For example, for a pair (r^i, r^j) of recipes described each by the ingredients they require, the representation of the variation is denoted by V^{ij}, where V^{ij} represents the variation of ingredients between r^i and r^j. Each ingredient ing is marked by $-$, $=$, or $+$:

- $ing^- \in V^{ij}$ if ing is an ingredient of r^i but not of r^j.
- $ing^+ \in V^{ij}$ if ing is an ingredient of r^j but not of r^i.
- $ing^= \in V^{ij}$ if ing is an ingredient of both r^i and r^j.

Table 2 shows the ingredient variations for 3 ordered pairs of recipes described in Table 1. An AK discovery process based on FCIs can be run on such a binary table which constitutes a formal context. Each extracted FCI produces an adaptation rule ar with a support supp(ar), i.e. the number of V^{ij} containing ar. For example, for $\sigma = 2$, $\{$PieCrust$-$, PuffPastry$+\}$ is an FCI which produces an adaptation rule consisting in replacing the pie crust by a puff pastry.

2.3 Assumptions and notations about CBR

Let $\mathcal{P}$ and $\mathcal{S}$ be two sets. A *problem* (resp., a *solution*) is an element of $\mathcal{P}$ (resp., of $\mathcal{S}$). The existence of a binary relation with the semantics "has for solution" is assumed. In this paper, this relation is assumed to be functional, guided by the idea to fully automatize the evaluation process; let f be the function from $\mathcal{P}$ to $\mathcal{S}$ such that $\mathtt{y} = \mathtt{f}(\mathtt{x})$ if y is the solution of x. A *case* is a pair $(\mathtt{x}, \mathtt{y}) \in \mathcal{P} \times \mathcal{S}$ where $\mathtt{y} = \mathtt{f}(\mathtt{x})$.

A CBR system on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ is built with a knowledge base $\mathtt{KB} = (\mathtt{CB}, \mathtt{DK}, \mathtt{RK}, \mathtt{AK})$ where CB is a finite set of cases, DK is the domain knowledge, RK is the retrieval knowledge (in this work, $\mathtt{RK} = \mathtt{dist}$, a distance function on $\mathcal{P}$), and AK is the adaptation knowledge that will take the form of adaptation rules.

A CBR system on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ aims at associating to a query problem $\mathtt{x}^{\mathtt{tgt}}$ a $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$, denoted by $\mathtt{y}^{\mathtt{tgt}} = f_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}})$. The function f_{CBR} is intended to be an approximation of f. It is built thanks to the following functions:

- the retrieval function, with the profile $\mathtt{retrieval} : \mathtt{x}^{\mathtt{tgt}} \mapsto (\mathtt{x}^s, \mathtt{y}^s) \in \mathtt{CB}$;
- the adaptation function, with the profile $\mathtt{adaptation} : ((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \mapsto \mathtt{y}^{\mathtt{tgt}} \in \mathcal{S}$; it is usually based on DK and AK. $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ is an *adaptation problem*.

Thus $f_{\mathrm{CBR}}(\mathtt{x}^{\mathtt{tgt}}) = \mathtt{adaptation}(\mathtt{retrieval}(\mathtt{x}^{\mathtt{tgt}}), \mathtt{x}^{\mathtt{tgt}})$.

With no domain and adaptation knowledge ($\mathtt{DK} = \mathtt{AK} = \emptyset$), the adaptation consists usually of a mere copy of the solution. This process is called *null adaptation*:

$$\mathtt{null\_adaptation} : ((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \mapsto \mathtt{y}^s$$

Adaptation principle using adaptation rules. Generally speaking, an adaptation rule ar is a function mapping an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \in \mathtt{CB} \times \mathcal{P}$ to $\mathtt{y}^{\mathtt{tgt}} \in \mathcal{S} \cup \{\mathtt{failure}\}$. Two cases of failure ($\mathtt{y}^{\mathtt{tgt}} = \mathtt{failure}$) are considered: (i) no $(\mathtt{x}^s, \mathtt{y}^s) \in \mathtt{CB}\}$ such as $\mathtt{dist}(\mathtt{x}^s, \mathtt{x}^{\mathtt{tgt}}) \leq \sigma$, with σ a given threshold, is returned by the retrieval function, and (ii) no AR ar is applicable on this adaptation problem. Else, $\mathtt{y}^{\mathtt{tgt}}$ is a proposed solution to $\mathtt{x}^{\mathtt{tgt}}$, by adaptation of $(\mathtt{x}^s, \mathtt{y}^s)$ according to ar. A score $\mathtt{supp}(\mathtt{ar}) \geq 0$ is associated with a rule ar; the higher is $\mathtt{supp}(\mathtt{ar})$, the more ar is preferred.

The adaptation consists in selecting the subset AAR of AK of applicable adaptation rules with maximum support: $\mathtt{ar} \in \mathtt{AAR}$ iff $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) \neq \mathtt{failure}$ and there exists no $\mathtt{ar}' \in \mathtt{AAR}$ such that $\mathtt{supp}(\mathtt{ar}') > \mathtt{supp}(\mathtt{ar})$.

2.4 Boolean setting

We presented at the beginning of section 2 the interest of AK discovery for a real life application. However, evaluating how an AK approach improves the results of the CBR system in such a concrete application is difficult because experiments require humans (users or experts) who have to evaluate the quality or the validity of the system answers (see for example [8] where the evaluation process implies users who have to judge if the TAAABLE cooking system returns correct or incorrect answers in the context of collaborative knowledge acquisition). Such a human based evaluation has many inconvenients: (a) specific interfaces have to be built to guarantee a blind evalution, (b) it is time consuming, especially comparing to an automated evaluation, (c) it requires available users and/or experts, and (d) the coverage of the evaluation is limited because of points (b) and (c).

For these reasons, we use in this work a Boolean setting in which all experiments can be automatized using Boolean functions as f. Let $\mathbb{B} = \{0, 1\}$ be the set of Boolean values. The Boolean operators are denoted by the connector symbols of propositional logic: for $a, b \in \mathbb{B}$, $\neg a = 1 - a$, $a \wedge b = \min(a, b)$, $a \vee b = \max(a, b)$, $a \oplus b = |b - a|$ ($\oplus$ is the exclusive or) and $a \Leftrightarrow b = \neg(a \oplus b)$.

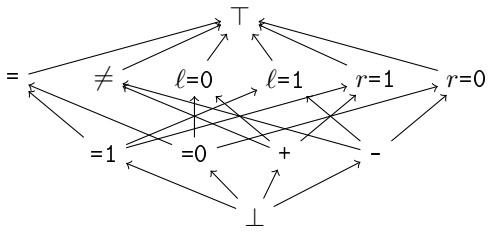

Fig. 1: The generalization/specialization hierarchy of variations.

Let $p \geq 0$. In the examples, an element of $\mathbb{B}^p$ is noted without parentheses and commas: $(0,1,0,0,1)$ is simply noted by 01001. The Hamming distance H on $\mathbb{B}^p$ is defined by $H(a,b) = \sum_{i=1}^{p} |b_i - a_i|$. For example, with $p = 5$, $H(01001, 11011) = 2$.

Let $m, n \in \mathbb{N}^*$, $\mathcal{P} = \mathbb{B}^m$, $\mathcal{S} = \mathbb{B}^n$ and $\mathtt{f} : \mathcal{P} \to \mathcal{S}$, be a boolean function to be approximated. A CBR system is considered on $(\mathcal{P}, \mathcal{S}, \mathtt{f})$ with $\mathtt{DK} = \emptyset$, $\mathtt{RK} = \mathtt{dist}$, the Hamming distance on $\mathcal{P}$, and AK a set of adaptation rules.

Adaptation rule language. The adaptation rule language used in this work is based on the notion of variations between Booleans, as described hereafter. Given $\ell, r \in \mathbb{B}$ (ℓ stands for *left*, r for *right*), the variation from ℓ to r is represented by *variation symbols*. Each of the 4 ordered pairs (ℓ, r) is represented by a primary variation symbol v:

- $(\ell, r) = (1, 0)$ is represented by $v = \mathtt{-}$;
- $(\ell, r) = (0, 1)$ is represented by $v = \mathtt{+}$;
- $(\ell, r) = (0, 0)$ is represented by $v = 0$;
- $(\ell, r) = (1, 1)$ is represented by $v = 1$.

Each primary variation symbols is linked to 3 inferred variation symbols. Fig. 1 shows the generalization links between the primary variation symbols and the inferred ones. For example, $v = \mathtt{-}$ is linked to $\neq$, stating that $\ell \neq r$, to ℓ=1 and to r=0, stating that ℓ (resp. r) is equal to 1 (resp. 0).

Given two cases $\mathtt{c}^1 = (\mathtt{x}^1, \mathtt{y}^1)$ and $\mathtt{c}^2 = (\mathtt{x}^2, \mathtt{y}^2)$, the variation V^{12} from $\mathtt{c}^1$ to $\mathtt{c}^2$ is encoded by the set of the expressions $\mathtt{x}_i^v$ and $\mathtt{y}_j^w$ such that v (resp., w) is a variation symbol from $\mathtt{x}_i^1$ to $\mathtt{x}_i^2$ (resp., from $\mathtt{y}_j^1$ to $\mathtt{y}_j^2$). For example, if $(\mathtt{x}^1, \mathtt{y}^1) = ((0,1),0)$ and $(\mathtt{x}^2, \mathtt{y}^2) = ((0,0),1)$ then $V^{12} = \{\mathtt{x}_1^{=0}, \mathtt{x}_1^{=}, \mathtt{x}_1^{\ell=0}, \mathtt{x}_1^{r=0}, \mathtt{x}_2^{-}, \mathtt{x}_2^{\neq}, \mathtt{x}_2^{\ell=1}, \mathtt{x}_2^{r=0}, \mathtt{y}_1^{+}, \mathtt{y}_1^{\neq}, \mathtt{y}_1^{\ell=0}, \mathtt{y}_1^{r=1}\}$.

An adaptation rule $\mathtt{ar}$ is a set of expressions $\mathtt{x}_i^v$ and $\mathtt{y}_j^w$. It is applicable on an adaptation problem $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ if there exists $\mathtt{y}^{\mathtt{tgt}} \in \mathbb{B}^n$ such that $V^{st} \supseteq \mathtt{ar}$ (where V^{st} represents the variation from $(\mathtt{x}^s, \mathtt{y}^s)$ to $(\mathtt{x}^{\mathtt{tgt}}, \mathtt{y}^{\mathtt{tgt}})$). If it is applicable, then its application consists in choosing such a $\mathtt{y}^{\mathtt{tgt}}$. If several $\mathtt{y}^{\mathtt{tgt}}$'s exist, the chosen one is the closest to $\mathtt{y}^s$ according to the Hamming distance on $\mathcal{S} = \mathbb{B}^n$, meaning that if $\mathtt{ar}$ gives no constraint on some $\mathtt{y}_j^{\mathtt{tgt}}$ then $\mathtt{y}_j^{\mathtt{tgt}} = \mathtt{y}_j^s$. For example:

if $\mathtt{ar} = \{\mathtt{x}_1^{-}, \mathtt{x}_2^{=}, \mathtt{y}_1^{+}\}$, $(\mathtt{x}^s, \mathtt{y}^s) = ((1,0,0),(0,0))$ and $\mathtt{x}^{\mathtt{tgt}} = (0,0,1)$
then $\mathtt{ar}$ is applicable on $((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}})$ and $\mathtt{ar}((\mathtt{x}^s, \mathtt{y}^s), \mathtt{x}^{\mathtt{tgt}}) = \mathtt{y}^{\mathtt{tgt}} = (1,0)$

3 AK discovery using positive and negative cases

An originality of this work is to build an AK discovey for CBR exploiting both *positive* and *negative* source cases. For a case $c = (\mathtt{x}, \mathtt{y}) \in \mathtt{CB}$, the case c is said *positive* if $\mathtt{y} = \mathtt{f}(\mathtt{x})$ and *negative* if $\mathtt{y} \neq \mathtt{f}(\mathtt{x})$. We denote $\mathtt{CB}^+$ (resp. $\mathtt{CB}^-$), the set of positive (resp. negative) cases of $\mathtt{CB}$, with $\mathtt{CB} = \mathtt{CB}^+ \cup \mathtt{CB}^-$.

Starting from the two sets of cases $\mathtt{CB}^+$ and $\mathtt{CB}^-$, ordered pairs of cases $(\mathtt{c}^1, \mathtt{c}^2)$ are formed, with $\mathtt{c}^1 = (\mathtt{x}^1, \mathtt{y}^1) \in \mathtt{CB}^+$ and $\mathtt{c}^2 = (\mathtt{x}^2, \mathtt{y}^2) \in \mathtt{CB}$ such that $\mathtt{x}^1 \neq \mathtt{x}^2$. Each such pair is encoded by a set V^{12} of the variations from $\mathtt{x}_i^1$ to $\mathtt{x}_i^2$ and from $\mathtt{y}_j^1$ to $\mathtt{y}_j^2$, as presented before. When $\mathtt{c}^2 \in \mathtt{CB}^+$, the variations between $\mathtt{c}^1$ and $\mathtt{c}^2$ can be considered as a positive example of AR (i.e. the application of the AR will produce a correct answer). When $\mathtt{c}^2 \in \mathtt{CB}^-$, the variations between $\mathtt{c}^1$ and $\mathtt{c}^2$ can be considered as a negative example of AR (i.e. the application of AR will produce an incorrect answer).

3.1 Exploiting positive examples

The AR learning process based on FCI extraction takes in input a set of V^{ij}, which will be used to build the formal context. In this work and especially for the evaluation, we have used two approaches to build the formal context. The first one consists in using each V^{ij} as an object with only the primary variations as properties. This approach will be noted AK^+ in the following. The second approach consists in extending AK^+ by using also the more general variations that can be inferred from the primary ones as object properties. This is a classical AK approach for extracting more general adaptation rules (i.e. rules with a higher support) (e.g. [6]). This second AK approach will be noted AK^{+I} in the following.

3.2 Exploiting negative examples

The objective of the AK^{+I} approach is the extraction of more general AR. However, when an AR is too general, its application is likely to give an incorrect answer. This is for example the case when the general rule consisting in replacing a pie crust by a puff pastry is applied to a salted tart recipe. This observation motivates the exploitation of negative examples for filtering too general ARs.

Exploiting negative examples in a learning process requires a specific approach. Some machine learning approaches such as, for example the version space model introduced by Mitchell [13], considers a training set composed of positive and negative examples in order to learn a binary classification model. The idea of the version space model is to build a space of hypotheses (view a disjunction of logical sentences) such that a hypothesis covers all positive examples and no negative ones. More recently, Ganter and Kuznetsov established the link between the version space model and FCA [9, 12]. Our exploitation of negative examples in order to extract ARs is based on the same idea introduced in these related works: generating AR covering positive examples without covering negative ones. A first approach to address this issue consists in generating AR only on a formal context built on positive examples and then removing rules which cover at least one negative example. Let V^{e^-} be the set of variations of the negative

example e^-, $\mathtt{ar}$ is removed if $e^- \supseteq \mathtt{ar}$. However, the complexity of this approach (in $O(|\mathrm{AR}| \times |\mathtt{CB}^-|)$) leads us to consider another more efficient way to compute ARs consistent with negative examples. For this, we take advantage of the efficiency of the FCI extraction algorithms in adding in the formal context the negative examples and by removing, before generating the AR, the CI which extent contains at least one negative example. This appproach exploiting both positive and negative examples (with inferred variations) will be noted AK^{+-I} in the following.

4 Evaluation

The objective of the evaluation is to study, on various types of Boolean functions, how exploiting negative cases in addition to positive ones improves the results of the CBR system. Experimental results are presented and discussed.

4.1 Experiment setting

In the experiment, $\mathcal{P} = \mathbb{B}^8$ and $\mathcal{S} = \mathbb{B}$, Functions $\mathtt{f}$ are randomly generated using the following generators that are based on the three main normal forms, with the purpose of having various types of functions:

`CNF f` is generated in a conjunctive normal form, i.e., $\mathtt{f}(\mathtt{x})$ is a conjunction of n_{conj} disjunctions of literals, for example $\mathtt{f}(\mathtt{x}) = (\mathtt{x}_1 \vee \neg\mathtt{x}_7) \wedge (\neg\mathtt{x}_3 \vee \mathtt{x}_7 \vee \mathtt{x}_8) \wedge \mathtt{x}_4$. The value of n_{conj} is randomly chosen uniformly in $\{3, 4, 5\}$. Each disjunction is generated on the basis of two parameters, $p^+ > 0$ and $p^- > 0$, with $p^+ + p^- < 1$: each variable $\mathtt{x}_i$ occurs in the disjunct in a positive (resp. negative) literal with a probability p^+ (resp., p^-). In the experiment, the values $p^+ = p^- = 0.1$ were chosen.

`DNF f` is generated in a disjunctive normal form, i.e., it has the same form as for `CNF` except that the connectors $\wedge$ and $\vee$ are exchanged. The parameters n_{disj}, p^+ and p^- are set in the same way.

`Pol` is the same as `DNF`, except that the disjunctions ($\vee$) are replaced with exclusive or's ($\oplus$), thus giving a polynomial normal form. The only different parameter is $p^- = 0$ (only positive literals occur in the polynomial normal form).

The case base $\mathtt{CB} = \mathtt{CB}^+ \cup \mathtt{CB}^-$ is generated randomly, with the values for their sizes:$|\mathtt{CB}^+| \in \{16, 32, 48\}$, i.e. $|\mathtt{CB}^+|$ is between $\frac{1}{16}$ and $\frac{3}{16}$ of $|\mathcal{P}| = 2^8 = 256$, and $|\mathtt{CB}^-| \in \{0, \frac{|\mathtt{CB}^+|}{2}, |\mathtt{CB}^+|\}$.

Each positive source case $(\mathtt{x}, \mathtt{y})$ is generated as follows: $\mathtt{x}$ is randomly chosen in $\mathcal{P}$ with a uniform distribution and $\mathtt{y} = \mathtt{f}(\mathtt{x})$. Each negative source case $(\mathtt{x}_1, \mathtt{y}_1)$ is generated by generating a positive source case $(\mathtt{x}, \mathtt{y})$ with an inversion of the solution $\mathtt{y}$: $\mathtt{y}_1 = \neg\mathtt{y}$.

Five adaptation approaches are tested: AK^+, AK^{+-}, AK^{+I}, AK^{+-I} and NN, the classical nearest neighbour approach with the `null_adaptation` for adaptation function. The retrieve and adaptation processes attempt to adapt, for NN, the 3 source cases which are the closest to the target problem according to `dist` with a maximal distance of 2 on $\mathcal{P}$. For the three approaches based on AK, there is no threshold on the

		prec									car								
	$\lvert CB^+\rvert =$	16			32			48			16			32			48		
	$\lvert CB^-\rvert =$	0	8	16	0	16	32	0	24	48	0	8	16	0	16	32	0	24	48
CNF	NN	.82	.82	.82	.84	.84	.84	.84	.84	.84	.75	.75	.75	.84	.84	.84	.84	.84	.84
	AK^+	.81	.81	.80	.79	.79	.79	.78	.78	.79	.81	.81	.80	.79	.79	.79	.78	.78	.79
	AK^{+-}	.81	.85	.85	.79	.88	.91	.78	.90	.94	.81	.85	.85	.79	.88	.87	.78	.85	.83
	AK^{+I}	.78	.78	.78	.67	.67	.67	.76	.76	.76	.78	.78	.78	.67	.67	.67	.76	.76	.76
	AK^{+-I}	.78	.82	.84	.67	.84	.87	.76	.86	.92	.78	.82	.83	.67	.84	.86	.76	.86	.91
DNF	NN	.80	.80	.80	.82	.82	.82	.83	.83	.83	.74	.74	.74	.81	.82	.82	.83	.83	.83
	AK^+	.77	.77	.77	.76	.76	.76	.81	.81	.81	.77	.77	.77	.76	.76	.76	.81	.81	.81
	AK^{+-}	.77	.79	.83	.76	.85	.90	.81	.88	.94	.77	.79	.82	.76	.84	.82	.81	.85	.82
	AK^{+I}	.56	.56	.56	.66	.66	.66	.63	.63	.64	.56	.56	.56	.66	.66	.66	.63	.63	.64
	AK^{+-I}	.56	.72	.78	.66	.79	.87	.63	.85	.92	.56	.72	.77	.66	.79	.85	.63	.85	.91
POL	NN	.59	.59	.59	.63	.64	.64	.64	.64	.64	.54	.53	.54	.63	.63	.63	.64	.64	.64
	AK^+	.53	.54	.54	.56	.56	.56	.56	.55	.55	.53	.54	.54	.56	.56	.56	.56	.55	.55
	AK^{+-}	.53	.54	.61	.56	.77	.91	.56	.80	.93	.53	.53	.52	.56	.41	.34	.56	.30	.21
	AK^{+I}	.49	.50	.50	.44	.44	.44	.42	.42	.43	.49	.50	.50	.44	.44	.44	.42	.42	.43
	AK^{+-I}	.49	.52	.61	.44	.58	.72	.42	.68	.85	.49	.51	.56	.44	.53	.52	.42	.55	.51

Table 3: `prec` and `car` for the three generators for different case base sizes.

maximal distance: all source cases for which AR can be applied participate to solve the problem. A vote on the results computed from the retrieved cases is used to associate a unique element $\mathtt{y} \in \mathcal{S}$. Moreover, a vote is also used when using AR: 3 ARs with the higher supports are used to adapt each of the source cases and the most frequent result wins.

All the approaches are evaluated according to two measures: the precision `prec` and the correct answer rate `car`. Let `ntp` be the number of target problems posed to the system, `na` be the number of (correct or incorrect) answers ($\mathtt{ntp} - \mathtt{na}$ is the number of target problems for which the system fails to propose a solution), and `nca` be the number of correct answers (according to the generated function `f`) and the predicted answer. So, the precision `prec` is defined as the average of the ratios $\frac{\mathtt{nca}}{\mathtt{na}}$, and the correct answer rate `car` is defined as the average of the ratios $\frac{\mathtt{nca}}{\mathtt{ntp}}$.

The average is computed on 10 `f` for each of the 3 function generators. With the different sizes for $\mathtt{CB}^+$ and $\mathtt{CB}^-$, 1872 cases are solved for each `f`.

4.2 Results and discussion

Table 3 presents the precision and correct answer rate of the five approaches for the differents function generators with various sizes of $\mathtt{CB}^+$ and $\mathtt{CB}^-$. The results show that exploiting negative cases improves the CBR system results both when using inferences and not (comparison of AK^+ wrt $AK^{+-}$and AK^{+I} wrt AK^{+-I}).

5 Conclusion

*** todo ***

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications **7**(1) (March 1994) 39–59
2. Badra, F., Cordier, A., Lieber, J.: Opportunistic Adaptation Knowledge Discovery. In McGinty, L., Wilson, D.C., eds.: 8th International Conference on Case-Based Reasoning - ICCBR 2009. Volume 5650 of Lecture Notes in Computer Science., Seattle, États-Unis, Springer (2009) 60–74
3. Berasaluce, S., Laurencco, C., Napoli, A., Niel, G.: An Experiment on Mining Chemical Reaction Databases. In Le Thi, H.A., Dinh, T.P., eds.: Modelling, Computation and Optimization in Information Systems and Management Sciences - MCO'04, Metz, France, Hermes Science Publishing, London (2004) 535–542
4. Cordier, A., Dufour-Lussier, V., Lieber, J., Nauer, E., Badra, F., Cojan, J., Gaillard, E., Infante-Blanco, L., Molli, P., Napoli, A., Skaf-Molli, H.: Taaable: a Case-Based System for personalized Cooking. In Montani, S., Jain, L.C., eds.: Successful Case-based Reasoning Applications-2. Volume 494 of Studies in Computational Intelligence. Springer (2014) 121–162
5. d'Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A., Szathmary, L.: Case base mining for adaptation knowledge acquisition. In Veloso, M.M., ed.: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07), Morgan Kaufmann, Inc. (2007) 750–755
6. Dufour-Lussier, V., Lieber, J., Nauer, E., Toussaint, Y.: Improving case retrieval by enrichment of the domain ontology. In: 19th International Conference on Case Based Reasoning - ICCBR'2011, London, United Kingdom (2011)
7. Gaillard, E., Lieber, J., Nauer, E.: Adaptation knowledge discovery for cooking using closed itemset extraction. In: The Eighth International Conference on Concept Lattices and their Applications - CLA 2011, Nancy, France (2011)
8. Gaillard, E., Lieber, J., Nauer, E.: How Managing the Knowledge Reliability Improves the Results of a Reasoning Process. In: 16th European Conference on Knowledge Management - ECKM 2015, Udine, Italy (2015) 10 pages
9. Ganter, B., Kuznetsov, S.O.: Hypotheses and version spaces. In Ganter, B., de Moor, A., Lex, W., eds.: Conceptual Structures for Knowledge Creation and Communication, Berlin, Heidelberg, Springer Berlin Heidelberg (2003) 83–95
10. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
11. Hanney, K., Keane, M.T.: Learning adaptation rules from a case-base. In Smith, I., Faltings, B., eds.: Advances in Case-Based Reasoning – Third Eur. Workshop, EWCBR'96. LNAI 1168, Springer Verlag, Berlin (1996) 179–192
12. Kuznetsov, S.O.: Complexity of learning in concept lattices from positive and negative examples. Discrete Applied Mathematics **142**(1) (2004) 111 – 125
13. Mitchell, T.: Version Space: An Approach to Concept Learning,. PhD thesis, Stanford University (1978)
14. Richter, M.M., Weber, R.O.: Case-based reasoning, a textbook. Springer (2013)
15. Skaf-Molli, H., Desmontils, E., Nauer, E., Canals, G., Cordier, A., Lefevre, M., Molli, P., Toussaint, Y.: Knowledge Continuous Integration Process (K-CIP). In: 21st WWW Conference - Semantic Web Collaborative Spaces workshop, Lyon, France (2012) 1075–1082

16. Szathmary, L., Napoli, A.: CORON: A Framework for Levelwise Itemset Mining Algorithms. Supplementary Proc. of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France (2005) 110–113